Kalman Temporal Differences: the deterministic case

Matthieu Geist, Olivier Pietquin and Gabriel Fricout

***Abstract*— This paper deals with value function and Q-function approximation in deterministic Markovian decision processes. A general statistical framework based on the Kalman filtering paradigm is introduced. Its principle is to adopt a parametric representation of the value function, to model the associated parameter vector as a random variable and to minimize the mean-squared error of the parameters conditioned on past observed transitions. From this general framework, which will be called *Kalman Temporal Differences* (KTD), and using an approximation scheme called the unscented transform, a family of algorithms is derived, namely KTD-V, KTD-SARSA and KTD-Q, which aim respectively at estimating the value function of a given policy, the Q-function of a given policy and the optimal Q-function. The proposed approach holds for linear and nonlinear parameterization. This framework is discussed and potential advantages and shortcomings are highlighted.**

I. Introduction

THIS paper deals with value function and Q-function approximation in deterministic Markovian Decision Process (MDP). An MDP is a tuple $\{S, A, T, R, \gamma\}$, where S is the state space, A the action space, $T : S \times A \to S$ the deterministic transition function, $R : S \times A \times S \to \mathbb{R}$ the bounded reward function, and γ the discount factor. A policy π is a (here deterministic) mapping from states to actions, $\pi : S \to A$. The value function of a given policy is classically defined as:

$$V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi] \tag{1}$$

where r_i is the reward observed at time i. The Q-function is defined as:

$$Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi] \tag{2}$$

Reinforcement learning (RL) [1] aims at finding (through interaction) the policy π^* which maximises the value function for every state:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}(V^\pi) \tag{3}$$

Two schemes (among others) can lead to the solution. First, policy iteration implies to learn the value function of a given policy, and then improve the policy, the new one being greedy respectively to the learned value function. It implies to solve the *Bellman evaluation equation*, which is given here for the value function and the Q-function, respectively:

$$V^\pi(s) = R(s, \pi(s), s') + \gamma V^\pi(s'), \forall s \tag{4}$$

$$Q^\pi(s,a) = R(s,a,s') + \gamma Q^\pi(s', \pi(s')), \forall s, a \tag{5}$$

M. Geist and O. Pietquin are with the IMS Research Group, Supélec, Metz, France (email: {matthieu.geist, olivier.pietquin}@supelec.fr). M. Geist and G. Fricout are with the MC cluster, ArcelorMittal Research, Maizières-lès-Metz, France. M. Geist is also with the CORIDA project-team, INRIA Nancy-Grand Est, France.

Here and through the rest of the paper, s' denotes the transiting state, that is $s' = T(s, \pi(s))$ or $s' = T(s,a)$, depending on the context. The second scheme, called value iteration, aims directly at finding the optimal policy. It implies to solve the *Bellman optimality equation*, which is given here for the Q-function:

$$Q^*(s,a) = R(s,a,s') + \gamma \max_{b \in A} Q^*(s', b), \forall s, a \tag{6}$$

The aim of this paper is to find an approximate solution of the Bellman evaluation or optimality equations, for the value function or the Q-function, when the state or action spaces are too large for classical dynamic programming or reinforcement learning algorithms [2]. Moreover the proposed algorithm is online, so as to keep this important RL feature.

To do so, Temporal Differences (TD) algorithms will be considered. They form a class of methods which consist in correcting the representation of the value (or Q-) function according to the TD error made on it. Most of them can be generically written as:

$$\theta_{i+1} = \theta_i + K_i \delta_i \tag{7}$$

In this expression, θ_i is the current representation of the value function, θ_{i+1} is an updated representation given an observed transition, δ_i is the so-called temporal difference error, and K_i is a gain indicating in which direction the representation of the value function should be corrected. Each of these terms will now be discussed.

If the state space S and the action space A are finite and small enough, an exact description of the value function is possible, and θ will be a vector with as many components as the state (-action) space (tabular representation). If these spaces are too large, approximation is necessary. A classical choice in RL is the linear parameterization, that is the value function is approximated by:

$$\hat{V}_\theta(s) = \sum_{j=1}^{p} w_j \phi_j(s) \tag{8}$$

where $(\phi_j)_{1 \leq j \leq p}$ is a set of basis functions, which should be defined beforehand, and the weights w_j are the parameters:

$$\theta = [w_1, \ldots, w_p]^T \tag{9}$$

Many function approximation algorithms require such a representation to ensure convergence [3], or even to be applicable [4]. Other representations are possible such as neural networks where θ contains the set of associated synaptic weights. The proposed KTD framework is applicable to any representation of the value (or Q-) function, as long as it can be fully described by a finite set of p parameters.

978-1-4244-2761-1/09/$25.00 ©2009 IEEE

In (7), the term δ_i is the TD error. Suppose that at step i a transition (s_i, a_i, s_{i+1}, r_i) is observed. For TD-like RL algorithms, that is algorithms which aim at evaluating the value function of a given policy π, the TD error is:

$$\delta_i = r_i + \gamma \hat{V}_{\theta_i}(s_{i+1}) - \hat{V}_{\theta_i}(s_i) \tag{10}$$

For SARSA-like algorithms, that is algorithms which aim at evaluating the Q-function of a given policy π, the TD error is:

$$\delta_i = r_i + \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) - \hat{Q}_{\theta_i}(s_i, a_i) \tag{11}$$

Finally, for Q-learning-like algorithms, that is algorithms which aim at computing the optimal Q-function, the TD error is:

$$\delta_i = r_i + \gamma \max_{b \in A} \hat{Q}_{\theta_i}(s_{i+1}, b) - \hat{Q}_{\theta_i}(s_i, a_i) \tag{12}$$

The type of temporal difference which is used determines which Bellman equation is to be solved (evaluation equation for (10) and (11), optimality equation for (12)), and thus if the algorithm belongs to the policy iteration or value iteration family.

The term K_i is a gain which is specific to each algorithm. The most common are reviewed here. For TD, SARSA and Q-learning (see [1] for example), the gain can be written as

$$K_i = \alpha_i e_i \tag{13}$$

where α_i is a classical learning rate in stochastic approximation theory, and should verify:

$$\sum_{i=0}^{\infty} \alpha_i = \infty \text{ and } \sum_{i=0}^{\infty} \alpha_i^2 < \infty \tag{14}$$

and e_i is a unitary vector which is zero everywhere except in the component corresponding to the state s_i (or to the state-action (s_i, a_i)) which is equal to one (Kronecker function). These algorithms have been modified to consider so-called eligibility traces (see [1]), and the gain can then be written as

$$K_i = \alpha_i \sum_{j=1}^{i} \lambda^{i-j} e_j \tag{15}$$

where λ is the eligibility factor. These algorithms have also been extended to take into account approximate representation of the value function. According to [5] they are called direct algorithms. Without eligibility traces, the gain can be written as

$$K_i = \alpha_i \nabla_{\theta_i} \hat{V}_{\theta_i}(s_i) \tag{16}$$

where $\nabla_{\theta_i} \hat{V}_{\theta_i}(s_i)$ is the derivation following the parameter vector of the parameterized value function in the current state. The value function can be replaced straightforwardly by the Q-function in this gain. The direct algorithms have also been extended to take into account eligibility traces, which leads to the following gain:

$$K_i = \alpha_i \sum_{j=1}^{i} \lambda^{i-j} \nabla_{\theta_i} \hat{V}_{\theta_i}(s_j) \tag{17}$$

Another well known approach is the set of residual algorithms [5], for which the gain is obtained through the minimization of the L_2-norm of the Bellman residual (*i.e.* the difference between the left side and the right side of the Bellman equation):

$$K_i = \alpha_i \nabla_{\theta_i} \left(\hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \right) \tag{18}$$

The last approach we review is the Least-Squares Temporal Differences (LSTD) algorithm [4], which is only defined for linear parameterization (8) and for which the gain is defined recursively:

$$K_i = \frac{C_{i-1}\phi(s_i)}{1 + (\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1} \phi(s_i)} \tag{19}$$

$$C_i = C_{i-1} - \frac{C_{i-1}\phi(s_i)(\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1}}{1 + (\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1} \phi(s_i)} \tag{20}$$

where $\phi(s)$ is defined in (47). This algorithm has also been extended to eligibility traces, see [6] for details.

The problem addressed in this paper can be stated as: given a representation of the value function (or of the Q-function) summarized by the parameter vector θ and given a temporal difference scheme (or equivalently given a Bellman equation to be solved), what is the best gain K ? To answer this question, a statistical point of view is adopted here and the Kalman filtering framework [7] is followed. The proposed approach can be linked to papers based on Gaussian processes [8], least-squares [4] or Kalman filtering [9], [10]. This will be further discussed in Section VI. In the next section the general *Kalman Temporal Differences* (KTD) framework is presented. The following sections specialize it to derive a family of algorithms for the value function and the Q-function evaluation (KTD-V and KTD-SARSA), and for the Q-function optimization (KTD-Q). An approximating scheme, the so-called *unscented transform* [11], which is necessary to handle nonlinear parameterization and the Bellman optimality equation, is also presented. Eventually some points are discussed and perspectives are presented.

II. KTD: THE GENERAL CASE

In this section a very general point of view is adopted, and practical algorithms will be derived later. For now, a transition is generically noted as:

$$t_i = \begin{cases} (s_i, s_{i+1}) \\ (s_i, a_i, s_{i+1}, a_{i+1}) \\ (s_i, a_i, s_{i+1}) \end{cases} \tag{21}$$

given that the aim is the value function evaluation, the Q-function evaluation or the Q-function optimization. Similarly, for the same cases, the following shortcuts hold:

$$g_{t_i}(\theta_i) = \begin{cases} \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\theta_i}(s_{i+1}, b) \end{cases} \tag{22}$$

Thus all TD schemes can be written generically as

$$\delta_i = r_i - g_{t_i}(\theta_i) \tag{23}$$

As said before, a statistical point of view is adopted. The parameter vector θ_i is modeled as a random variable following a random walk. The problem at sight can then be stated in a so-called *state-space formulation*:

$$\begin{cases} \theta_{i+1} = \theta_i + v_i \\ r_i = g_{t_i}(\theta_i) + n_i \end{cases} \tag{24}$$

This expression is fundamental for the proposed framework. The first equation is the evolution equation, it specifies that the parameter vector follows a random walk which expectation corresponds to the optimal estimation of the value function. The evolution noise v_i is centered, white and independent. Notice that this allows handling non-stationary MDPs. The second equation is the observation equation, it links the observed transition to the value (or Q-) function through a Bellman equation. The observation noise n_i is supposed centered, white and independent. Notice that this necessary assumption does not hold for stochastic MDPs (see [12] for an expression of this noise in the stochastic case), that is why deterministic transitions are supposed here. This model noise arises from the fact that the solution of the Bellman equation does not necessarily exists in the functional space spanned by the set of parameter vectors.

The objective could be to estimate the parameter vector which minimizes the expectation of the mean-squared error conditioned on past observed rewards. The associated cost can be written as:

$$J(\hat{\theta}_i) = E\left[\|\theta_i - \hat{\theta}_i\|^2 | r_{1:i}\right] \text{ with } r_{1:i} = r_1, \ldots, r_i \tag{25}$$

Generally speaking, the minimum mean square error (MMSE) estimator is the conditional expectation:

$$\underset{\hat{\theta}_i}{\operatorname{argmin}}\, J(\hat{\theta}_i) = \hat{\theta}_{i|i} = E\left[\theta_i | r_{1:i}\right] \tag{26}$$

However, except in specific cases, this estimator is not computable. Instead, the aim is here to find the best *linear* estimator. It can be written in a form quite similar to equation (7):

$$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \tilde{r}_i \tag{27}$$

In equation (27), $\hat{\theta}_{i|i}$ is the estimate at time i, $\hat{\theta}_{i|i-1} = E[\theta_i | r_{1:i-1}]$ is the prediction of this estimate according to past observed rewards $r_{1:i-1}$, and for a random walk model the following equality holds:

$$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1} \tag{28}$$

The innovation

$$\tilde{r}_i = r_i - \hat{r}_{i|i-1} \tag{29}$$

is the difference between the observed reward r_i and its prediction based on the previous estimate of the parameter vector:

$$\hat{r}_{i|i-1} = E\left[g_{t_i}(\theta_i) | r_{1:i-1}\right] \tag{30}$$

Note that the innovation $\tilde{r}_i$ is not exactly the temporal difference defined in equation (23), which is a random variable through its dependency to the random vector θ_i. It is its expectation conditioned on past observed data.

Using classical equalities, the cost function can be rewritten as:

$$\begin{aligned} J(\hat{\theta}_i) &= E\left[\|\theta_i - \hat{\theta}_i\|^2 | r_{1:i}\right] \\ &= E\left[(\theta_i - \hat{\theta}_i)^T (\theta_i - \hat{\theta}_i) | r_{1:i}\right] \\ &= \operatorname{trace}\left(E\left[(\theta_i - \hat{\theta}_i)(\theta_i - \hat{\theta}_i)^T | r_{1:i}\right]\right) \\ &= \operatorname{trace}\left(\operatorname{cov}\left(\theta_i - \hat{\theta}_i | r_{1:i}\right)\right) \end{aligned} \tag{31}$$

A first step to the computation of the optimal gain is to express the conditioned covariance over parameters as a function of the gain K_i. A few more notations are first introduced (recall also the definition of the innovation (29)):

$$\tilde{\theta}_{i|i} = \theta_i - \hat{\theta}_{i|i} \text{ and } \tilde{\theta}_{i|i-1} = \theta_i - \hat{\theta}_{i|i-1} \tag{32}$$

$$P_{i|i} = \operatorname{cov}\left(\tilde{\theta}_{i|i} | r_{1:i}\right) \tag{33}$$

$$P_{i|i-1} = \operatorname{cov}\left(\tilde{\theta}_{i|i-1} | r_{1:i-1}\right) \tag{34}$$

$$P_{r_i} = \operatorname{cov}\left(\tilde{r}_i | r_{i:i-1}\right) \tag{35}$$

$$P_{\theta r_i} = E\left[\tilde{\theta}_{i|i-1} \tilde{r}_i | r_{1:i-1}\right] \tag{36}$$

Using the postulated update of equation (27), and the various estimators being unbiased, the covariance can be expanded:

$$\begin{aligned} P_{i|i} &= \operatorname{cov}\left(\theta_i - \hat{\theta}_{i|i} | r_{1:i}\right) \\ &= \operatorname{cov}\left(\theta_i - \left(\hat{\theta}_{i|i-1} + K_i \tilde{r}_i\right) | r_{1:i-1}\right) \\ &= \operatorname{cov}\left(\tilde{\theta}_{i|i-1} - K_i \tilde{r}_i | r_{1:i-1}\right) \\ P_{i|i} &= P_{i|i-1} - P_{\theta r_i} K_i^T - K_i P_{\theta r_i}^T + K_i P_{r_i} K_i^T \end{aligned} \tag{37}$$

The optimal gain can thus be obtained by deriving the trace of this matrix.

First note that the gradient being linear, for three matrices of *ad hoc* dimensions A, B and C, B being symmetric, the following algebraic identities hold:

$$\nabla_A \left(\operatorname{trace}\left(ABA^T\right)\right) = 2AB \tag{38}$$

$$\nabla_A \left(\operatorname{trace}\left(AC^T\right)\right) = \nabla_A \left(\operatorname{trace}\left(CA^T\right)\right) = C \tag{39}$$

and thus using equation (37) and previous identities:

$$\begin{aligned} & \nabla_{K_i} \left(\operatorname{trace}\left(P_{i|i}\right)\right) = 0 \\ \Leftrightarrow \quad & 2K_i P_{r_i} - 2P_{\theta r_i} = 0 \\ \Leftrightarrow \quad & K_i = P_{\theta r_i} P_{r_i}^{-1} \end{aligned} \tag{40}$$

Using (37) and (40), the covariance matrix $P_{i|i}$ is

$$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T \tag{41}$$

Notice that no Gaussian assumption has been made to derive these equations.

The most general KTD algorithm, which breaks down in three stages, can now be derived. The first step consists in computing predicted quantities $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$. Recall

that for a random walk model, equation (28) holds, and the predicted covariance can also be computed analytically:

$$\begin{aligned} P_{i|i-1} &= \text{cov}\left(\tilde{\theta}_{i|i-1}|r_{1:i-1}\right) \\ &= \text{cov}\left(\tilde{\theta}_{i-1|i-1} + v_{i-1}|r_{1:i-1}\right) \\ &= P_{i-1|i-1} + P_{v_{i-1}} \end{aligned} \tag{42}$$

where $P_{v_{i-1}}$ is the variance matrix of the evolution noise (which is given).

The second step is to compute some statistics of interest. It will be specialized for each algorithm of the next sections. The first statistic to compute is the prediction $\hat{r}_{i|i-1}$ (30). The second statistic to compute is the covariance between the parameter vector and the innovation:

$$P_{\theta r_i} = E\left[(\theta_i - \hat{\theta}_{i|i-1})(r_i - \hat{r}_{i|i-1})|r_{1:i-1}\right] \tag{43}$$

However, from the state-space model (24), $r_i = g_{t_i}(\theta_i) + n_i$, and the observation noise is centered and independent, so

$$P_{\theta r_i} = E\left[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})|r_{1:i-1}\right] \tag{44}$$

The last statistic to compute is the covariance of the innovation, which can be written (using again the characteristics of the observation noise):

$$\begin{aligned} P_{r_i} &= E\left[(r_i - \hat{r}_{i|i-1})^2|r_{1:i-1}\right] \\ &= E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1} + n_i)^2|r_{1:i-1}\right] \\ &= E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})^2|r_{1:i-1}\right] + P_{n_i} \end{aligned} \tag{45}$$

where P_{n_i} is the variance of the observation noise (which is also known).

The last step of the algorithm is the correction step. It consists in computing the gain (40), correcting the parameter vector (27) and updating the associated covariance matrix (41) accordingly. Notice that as the proposed method is online, it must thus be initialized with some priors $\hat{\theta}_{0|0}$ and $P_{0|0}$. The proposed general framework is summarized in Algorithm 1. The main difficulty in applying the KTD is to compute the statistics of interest $\hat{r}_{i|i-1}$, $P_{\theta r_i}$ and P_{r_i} (for which statistics $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ are necessary), which will be the subject of the next three sections.

Algorithm 1: General KTD algorithm

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition t_i and reward r_i ;

Prediction step;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Compute statistics of interest;
$\hat{r}_{i|i-1} = E[g_{t_i}(\theta_i)|r_{1:i-1}]$;
$P_{\theta r_i} = E\left[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_i)|r_{1:i-1}\right]$;
$P_{r_i} = E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})^2|r_{1:i-1}\right] + P_{n_i}$;

Correction step;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i\left(r_i - \hat{r}_{i|i-1}\right)$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

III. KTD-V

This section focuses on evaluating the value function, that is finding an approximate solution of the Bellman evaluation equation (4). First the linear case is considered, which allows deriving an analytical solution to statistics of interest computation. An approximation scheme, the unscented transform [11], is then introduced, which proves itself to be useful for the nonlinear parameterization case.

A. Linear parameterization

In this section the linear parameterization of equation (8) is adopted, which is shortened as:

$$V_\theta(s) = \phi(s)^T \theta \tag{46}$$

where the parameter vector is given in equation (9) and where $\phi(s)$ is a vector defined as:

$$\phi(s) = [\phi_1(s), \ldots, \phi_p(s)]^T \tag{47}$$

The state-space formulation (24) can thus be rewritten as:

$$\begin{cases} \theta_{i+1} = \theta_i + v_i \\ r_i = \left(\phi(s_i) - \gamma\phi(s_{i+1})\right)^T \theta_i + n_i \end{cases} \tag{48}$$

To shorten the equations, H_i is defined as:

$$H_i = \phi(s_i) - \gamma\phi(s_{i+1}) \tag{49}$$

As the observation equation is linear, the statistics of interest can be derived analytically. The prediction is

$$\begin{aligned} \hat{r}_{i|i-1} &= E\left[g_{t_i}(\theta_i)|r_{1:i-1}\right] \\ &= E\left[H_i^T \theta_i|r_{1:i-1}\right] \\ &= H_i^T E\left[\theta_i|r_{1:i-1}\right] \\ &= H_i^T \hat{\theta}_{i|i-1} \end{aligned} \tag{50}$$

The covariance between the parameter vector and the innovation can also be computed analytically:

$$\begin{aligned} P_{\theta r_i} &= E\left[\tilde{\theta}_{i|i-1}\left(g_{t_i}(\theta_i) - \hat{r}_{i|i-1}\right)|r_{1:i-1}\right] \\ &= E\left[\tilde{\theta}_{i|i-1} H_i^T \tilde{\theta}_{i|i-1}|r_{1:i-1}\right] \\ &= E\left[\tilde{\theta}_{i|i-1}\tilde{\theta}_{i|i-1}^T|r_{1:i-1}\right] H_i \\ &= P_{i|i-1} H_i \end{aligned} \tag{51}$$

The covariance of the innovation is derived analytically too:

$$\begin{aligned} P_{r_i} &= E\left[\left(g_{t_i}(\theta_i) - \hat{r}_{i|i-1}\right)^2|r_{1:i-1}\right] + P_{n_i} \\ &= E\left[\left(H_i^T \tilde{\theta}_{i|i-1}\right)^2|r_{1:i-1}\right] + P_{n_i} \\ &= H_i^T P_{i|i-1} H_i + P_{n_i} \end{aligned} \tag{52}$$

The gain can thus be defined algebraically and recursively:

$$K_i = \frac{P_{i|i-1}H_i}{H_i^T P_{i|i-1} H_i + P_{n_i}} \tag{53}$$

This gain shares similarities with the gain of the LSTD algorithm [4] (equation (19)), which is not a surprise. The LSTD is based on a least-squares minimization (however with the introduction of instrumental variables in order to handle stochastic transitions), and the Kalman filter can be seen as a generalization of the least-squares method. This gain also shares similarities with the one arising from a parametric Gaussian process modelling of the value function evaluation problem [8]. The KTD-V approach for linear parameterization is summarized in Algorithm 2.

Algorithm 2: KTD-V: linear parameterization

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition (s_i, s_{i+1}) and reward r_i ;

Prediction step;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Compute statistics of interest;
$\hat{r}_{i|i-1} = H_i^T \hat{\theta}_{i|i-1}$;
$P_{\theta r_i} = P_{i|i-1} H_i$;
$P_{r_i} = H_i^T P_{i|i-1} H_i + P_{n_i}$;
/* where $H_i = \phi(s_i) - \gamma\phi(s_{i+1})$ */

Correction step;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1}\right)$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

The next case to be addressed is the nonlinear parameterization of the value function. Basically, the issue of computing the statistics of interest for the KTD can be stated as the following problem: given the mean and covariance of a random variable, how can the mean and covariance of a nonlinear (and perhaps non derivable) mapping of this random variable be computed ? The following section presents the unscented transform, which is an approximation scheme designed to handle such a problem.

B. The Unscented Transform

Let's abstract a little bit from reinforcement learning and Kalman filtering. Let X be a random vector, and let Y be a mapping of X. The problem is to compute mean and covariance of Y knowing the mapping and first and second order moments of X. If the mapping is linear, the relation between X and Y can be written as $Y = AX$ where A is a matrix of *ad hoc* dimension. In this case, required mean and covariance can be analytically computed: $E[Y] = AE[X]$ and $E[YY^T] = AE[XX^T]A^T$. This result has been used to derive the KTD-V of Section III-A.

If the mapping is nonlinear, the relation between X and Y can be generically written as:

$$X = f(Y) \tag{54}$$

A first solution would be to approximate the nonlinear mapping, that is to linearize it around the mean of the random vector X. This leads to the following approximations of the mean and covariance of Y:

$$E[Y] \approx f\left(E[X]\right) \tag{55}$$

$$E[YY^T] \approx \left(\nabla f\left(E[X]\right)\right) E[XX^T] \left(\nabla f\left(E[X]\right)\right)^T \tag{56}$$

This approach is the base of Extended Kalman Filtering (EKF) [13], which has been extensively studied and used in past decades. However it has some limitations. First it cannot handle non-derivable nonlinearities, and thus cannot handle the Bellman optimality equation (6) because of the $\max$ operator. It requires to compute the gradient of the mapping f, which can be quite difficult even if possible. It also supposes that the nonlinear mapping is locally linearizable, which is unfortunately not always the case and can lead to quite bad approximation, as exemplified in [11].

The basic idea of unscented transform is that it is easier to approximate an arbitrary random vector than an arbitrary nonlinear function. Its principle is to sample *deterministically* a set of so-called sigma-points from the expectation and the covariance of X. The images of these points through the nonlinear mapping f are then computed, and they are used to approximate statistics of interest. It shares similarities with Monte-Carlo methods, however here the sampling is deterministic and requires less samples to be drawn, nonetheless allowing a given accuracy [11].

The original unscented transform is now described more formally (some variants have been introduced since, but the basic principle is the same). Let n be the dimension of X. A set of $2n+1$ sigma-points is computed as follows:

$$x_0 = \bar{X} \qquad j = 0 \tag{57}$$

$$x_j = \bar{X} + \left(\sqrt{(n+\kappa)P_X}\right)_j \qquad 1 \leq j \leq n \tag{58}$$

$$x_j = \bar{X} - \left(\sqrt{(n+\kappa)P_X}\right)_{n-j} \qquad n+1 \leq j \leq 2n \tag{59}$$

as well as associated weights:

$$w_0 = \frac{\kappa}{n+\kappa} \text{ and } w_j = \frac{1}{2\left(n+\kappa\right)} \forall j > 0 \tag{60}$$

where $\bar{X}$ is the mean of X, P_X is its variance matrix, κ is a scaling factor which controls the accuracy of the unscented transform [11], and $(\sqrt{(n+\kappa)P_X})_j$ is the j^{th} column of the Cholesky decomposition of the matrix $(n+\kappa)P_X$. Then the image through the mapping f is computed for each of these sigma-points:

$$y_j = f(x_j), \quad 0 \leq j \leq 2n \tag{61}$$

The set of sigma-points and their images can finally be used to compute first and second order moments of Y, and even

P_{XY}, the covariance matrix between X and Y:

$$\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y_j \tag{62}$$

$$P_Y \approx \sum_{j=0}^{2n} w_j \left(y_j - \bar{y}\right)\left(y_j - \bar{y}\right)^T \tag{63}$$

$$P_{XY} \approx \sum_{j=0}^{2n} w_j \left(x_j - \bar{x}\right)\left(y_j - \bar{y}\right)^T \tag{64}$$

where $\bar{x} = x_0 = \bar{X}$. The unscented transform having been presented, it is now possible to address the value function evaluation problem with nonlinear parameterization.

C. Nonlinear parameterization

In this section a generic parameterization of the value function $\hat{V}_\theta$ is considered: it can be a neural network [14], a parametric kernel representation [12], or any function representation of interest, as long as it can be described by a finite set of p parameters. The general state-space formulation (24) can thus be rewritten as:

$$\begin{cases} \theta_{i+1} = \theta_i + v_i \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i \end{cases} \tag{65}$$

The problem is still to compute the statistics of interest, which becomes tractable with the unscented transform. The first thing to compute is the set of sigma-points from known statistic $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ as described in Section III-B, as well as the associated weights:

$$\Theta_{i|i-1} = \left\{\hat{\theta}^{(j)}_{i|i-1},\ 0 \leq j \leq 2p\right\} \tag{66}$$

$$\mathcal{W} = \{w_j,\ 0 \leq j \leq 2p\} \tag{67}$$

Then the images of the sigma-points are computed, using the observation function of state-space model (65), which is linked to the Bellman evaluation equation (4):

$$\mathcal{R}_{i|i-1} = \left\{\hat{r}^{(j)}_{i|i-1} = \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i) - \gamma \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}),\ 0 \leq j \leq 2p\right\} \tag{68}$$

The sigma-points and their images being computed, the statistics of interest can be approximated by:

$$\hat{r}_{i|i-1} \approx \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1} \tag{69}$$

$$P_{r_i} \approx \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1}\right)^2 + P_{n_i} \tag{70}$$

$$P_{\theta r_i} \approx \sum_{j=0}^{2p} w_j \left(\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1}\right)\left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1}\right) \tag{71}$$

As the unscented transform is no longer an approximation for linear mapping, this formulation is still valid for value function evaluation with linear function approximation. The KTD-V with nonlinear function approximation is summarized in Algorithm 3.

Algorithm 3: KTD-V: nonlinear parameterization

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition (s_i, s_{i+1}) and reward r_i ;

Prediction Step;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Sigma-points computation ;
$\Theta_{i|i-1} = \left\{\hat{\theta}^{(j)}_{i|i-1},\quad 0 \leq j \leq 2p\right\}$;
$\mathcal{W} = \{w_j,\quad 0 \leq j \leq 2p\quad\}$;
$\mathcal{R}_{i|i-1} = \left\{\hat{r}^{(j)}_{i|i-1} = \right.$
$\left.\hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i) - \gamma \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}),\quad 0 \leq j \leq 2p\right\}$;

Compute statistics of interest;
$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
$P_{r_i} = \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1}\right)^2 + P_{n_i}$;

Correction step;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1}\right)$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

IV. KTD-SARSA

This section focuses on the Q-function evaluation of a given policy. The associated algorithm is called KTD-SARSA, which can be misleading. Indeed, generally speaking, SARSA is a Q-function evaluation algorithm associated with an optimistic policy iteration scheme. Here the focus is on the Q-function evaluation problem, and the control part is left apart. For a general parameterization $\hat{Q}_\theta$, and considering the Bellman evaluation equation (5), the state-space model (24) can be rewritten as:

$$\begin{cases} \theta_{i+1} = \theta_i + v_i \\ r_i = \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) + n_i \end{cases} \tag{72}$$

For a fixed policy, the value function evaluation on the state space induced Markov chain is quite similar to the Q-function evaluation on the state-action space induced Markov chain. It is thus straightforward to extend Algorithms 2 and 3 to Q-function evaluation. Recall that for a linear parameterization, the unscented transform leads to an exact computation of statistics of interest, and thus in this case Algorithm 3 is equivalent to Algorithm 2. That is why only the sigma-point formulation of KTD-SARSA is derived (Algorithm 4).

V. KTD-Q

This section focuses on the Q-function optimization, that is on finding an approximate solution of the Bellman optimality

Algorithm 4: KTD-SARSA

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition $(s_i, a_i, s_{i+1}, a_{i+1})$ and reward r_i;

Prediction Step;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Sigma-points computation ;
$\Theta_{i|i-1} = \left\{ \hat{\theta}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p \right\}$;
$\mathcal{W} = \{ w_j, \quad 0 \leq j \leq 2p \quad \}$;
$\mathcal{R}_{i|i-1} = \Big\{ \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) -$
$\gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}), \quad 0 \leq j \leq 2p \Big\}$;

Compute statistics of interest;
$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
$P_{r_i} = \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1} \right)^2 + P_{n_i}$;

Correction step;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1} \right)$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

Algorithm 5: KTD-Q

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition (s_i, a_i, s_{i+1}) and reward r_i ;

Prediction Step;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Sigma-points computation ;
$\Theta_{i|i-1} = \left\{ \hat{\theta}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p \right\}$;
$\mathcal{W} = \{ w_j, \quad 0 \leq j \leq 2p \quad \}$;
$\mathcal{R}_{i|i-1} = \Big\{ \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) -$
$\gamma \max_{b \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, b), \quad 0 \leq j \leq 2p \Big\}$;

Compute statistics of interest;
$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
$P_{r_i} = \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1} \right)^2 + P_{n_i}$;

Correction step;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1} \right)$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

equation (6). A general parameterization $\hat{Q}_\theta$ is adopted. The state-space model (24) can be specialized as follows:

$$\begin{cases} \theta_{i+1} = \theta_i + v_i \\ r_i = \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\theta_i}(s_{i+1}, b) + n_i \end{cases} \tag{73}$$

Here linear and nonlinear parameterization are not distinguished, because of the $\max$ operator, which is inherent to the Bellman optimality equation, and because of which the observation equation of state-space model (73) is nonlinear. This $\max$ operator is difficult to handle, especially because of its non-derivability.

Hopefully, as it approximates the random variable rather than the mapping, the unscented transform is a derivative free approximation. Given the general KTD algorithm introduced in Section II and the unscented transform described in Section III-B, it is possible to derive KTD-Q, the KTD algorithm for Q-function direct optimization. One has first to compute the set of sigma-point associated with the parameter vector, as in equations (66-67). Then the mapping of these sigma-points through the observation equation of state-space model (73), which contains the $\max$ operator, is computed:

$$\begin{aligned} \mathcal{R}_{i|i-1} = \Big\{ & \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) \\ & - \gamma \max_{b \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, b), \; 0 \leq j \leq 2p \Big\} \end{aligned} \tag{74}$$

Then, as usual, the sigma-points and their images are used to compute the statistics of interest, as in equations (69-71). The proposed KTD-Q is summarized in Algorithm 5.

VI. Discussion and Perspectives

A general Kalman-based function approximation scheme for reinforcement learning in deterministic Markovian decision processes has been introduced, and algorithms for value function and Q-function evaluation (policy iteration scheme) and for Q-function direct optimization (value iteration scheme) have been derived from it. Experimental results are not given here, however the KTD-Q algorithm has been first introduced in [12] from a Bayesian perspective, and related experiments are provided. Experimental results are promising.

As announced in Section I, related approaches have been proposed previously. In [8] a Gaussian process modelling of reinforcement learning function approximation is proposed. In the deterministic and parametric case, an algorithm which is almost the same as the KTD-V with linear parameterization is obtained (the only difference resides in the absence of process noise). However it is derived from a different point of view. As Kalman filtering is strongly linked to least-squares minimization, our approach shares similarities with the LSTD [4], however without taking into account instrumental variables concept. In [9] a Kalman filter designed to handle fixed-point approximation in the case of linear parameterization is introduced. It can be roughly seen as a bootstrapping version of the proposed KTD-V. Instead of the observation equation of state-space model (48), the following

observation equation is used:

$$r_i + \gamma\phi(s_{i+1})^T\hat{\theta}_{i-1|i-1} = \phi(s_i)^T\theta_i + n_i \qquad (75)$$

In other words, the reward is not considered as the observation, but an approximation of the value function is used to compute a "pseudo"-observation, and the update of the parameters is made so as to match the value function of the current state to this pseudo-observation. In [10], a bank of classical Kalman filters is used to learn the parameters of a piecewise linear parameterization of the value function. It can be roughly seen as a special case of the proposed approach, however differences exist: not one filter but a bank is used and the parameterization is piecewise linear, which is exploited to develop specificities of the algorithm.

The proposed framework has some potential advantages. First it does not suppose stationarity. An immediate application is to handle non-stationary environments. But an even more interesting one is the control case. The algorithm LSTD is known to fail when combined with optimistic policy iteration, because of the induced non-stationarities of this specific learning and control scheme. Kalman filtering and thus the proposed framework is designed to be robust to non-stationarities (random walk model of the parameter vector). This can be quite interesting for the control case, which has not been treated in this paper (the focus was on learning the value function or the Q-function, given observed transitions, and not on how to choose action for a given state). Second, the parameter vector is modelled as a random vector. As a consequence, at each time step, the covariance of this random vector is available. It can be propagated to the value function (as it is clearly a function of the parameters), using the unscented transform if necessary, in order to provide *local* uncertainty information for the value at a given state. This is exemplified in [15] for a simple regression problem. This uncertainty propagation can be useful to handle the well known dilemma between exploration and exploitation.

However, for now, the proposed framework presents a major drawback. In the case of stochastic transitions, the KTD can produce biased estimates of the parameters, or even be unstable. The problem lies in the fact that the KTD minimizes a squared Bellman residual (see [16] for the demonstration of the minimized cost function with unscented filtering and random walk evolution model):

$$J_i(\theta) = \sum_{j=1}^{i} P_{n_j}^{-1}\left(r_j - g_{t_j}(\theta)\right)^2 \qquad (76)$$

The cost function which should be considered to truly minimize the squared Bellman residual is

$$L(\theta) = \|V_\theta - TV_\theta\|^2 \qquad (77)$$

where T is one of the Bellman operators (depending on the Bellman equation to be solved and involving transition probabilities). As noted in [17], $J_i(\theta)$ is a biased estimator of $L(\theta)$, the bias being a variance term which favorises smooth value functions. The same problem arises in the residual approach [5]. A solution could be to introduce an auxiliary filter, in the same manner an auxiliary function has been introduced in [17]. For now, the KTD can be applied without modification to stochastic environments (see [12] for a successful application of KTD-Q to a stochastic problem), but it can become unstable, depending on the problem at sight.

To finish with, a new Kalman-based function approximation scheme for reinforcement learning has been introduced. Most interesting perspectives are to extend this framework to the control case, for which the non-stationarity hypothesis and the uncertainty propagation should be useful, and to handle more rigorously stochastic transitions. It is also planned to conduct more comparisons, theoretically and experimentally, of the KTD to other related function approximation schemes for reinforcement learning.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 3rd ed. The MIT Press, March 1998.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, May 1996.

[3] R. Schoknecht, "Optimality of Reinforcement Learning Algorithms with Linear Function Approximation," in *Conference on Neural Information Processing Systems (NIPS 15)*, 2002.

[4] S. J. Bradtke and A. G. Barto, "Linear Least-Squares Algorithms for Temporal Difference Learning," *Machine Learning*, vol. 22, no. 1-3, pp. 33–57, 1996.

[5] L. C. B. III, "Residual Algorithms: Reinforcement Learning with Function Approximation," in *International Conference on Machine Learning (ICML 95)*, 1995, pp. 30–37.

[6] J. A. Boyan, "Technical Update: Least-Squares Temporal Difference Learning," *Machine Learning*, vol. 49, no. 2-3, pp. 233–246, 1999.

[7] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[8] Y. Engel, "Algorithms and Representations for Reinforcement Learning," Ph.D. dissertation, Hebrew University, April 2005.

[9] D. Choi and B. V. Roy, "A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning," *Discrete Event Dynamic Systems*, vol. 16, pp. 207–239, 2006.

[10] C. W. Phua and R. Fitch, "Tracking Value Function Dynamics to Improve Reinforcement Learning with Piecewise Linear Function Approximation," in *International Conference on Machine Learning (ICML 07)*, 2007.

[11] S. J. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," in *Proceedings of the IEEE*, vol. 92, no. 3, March 2004, pp. 401–422.

[12] M. Geist, O. Pietquin, and G. Fricout, "Bayesian Reward Filtering," in *Proceedings of the European Workshop on Reinforcement Learning (EWRL 2008)*, ser. Lecture Notes in Artificial Intelligence, S. G. et al., Ed. Lille (France): Springer Verlag, June 2008, vol. 5323, pp. 96–109.

[13] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, 1st ed. Wiley & Sons, August 2006.

[14] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, USA: Oxford University Press, 1995.

[15] M. Geist, O. Pietquin, and G. Fricout, "A Sparse Nonlinear Bayesian Online Kernel Regression," in *Proceedings of the Second IEEE International Conference on Advanced Engineering Computing and Applications in Sciences (AdvComp 2008)*, Valencia (Spain), October 2008, pp. 199–204.

[16] R. van der Merwe, "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models," Ph.D. dissertation, OGI School of Science & Engineering, Oregon Health & Science University, Portland, OR, USA, April 2004.

[17] A. Antos, C. Szepesvári, and R. Munos, "Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path," *Machine Learning*, vol. 71, no. 1, pp. 89–129, April 2008.